NETWORK WORKING GROUP
REQUEST FOR COMMENTS #89
NIC 5697

BOB METCALFE
19 JAN 1971

SOME HISTORIC MOMENTS IN NETWORKING

While awaiting the completion of an interim network control program (INCP) for the MIT MAC Dynamic Modeling/Computer Graphics PDP-6/10 System (MITDG), we were able to achieve a number of 'historic moments in networking' worthy of some comment. First, we were able to connect an MITDG terminal to a Multics process making it a Multics terminal. Second, we successfully attached an MITDG terminal to the Harvard PDP-10 System thereby enabling automatic remote use of the Harvard System for MIT. Third, we developed primitive mechanisms through which remotely generated programs and data could be transmitted to our system, executed, and returned. Using these mechanisms in close cooperation with Harvard, we received graphics programs and 3D data from Harvard's PDP-10, processed them repeatedly using our Evans & Sutherland Line Drawing System (the E&S), and transmitted 2D scope data to Harvard's PDP-1 for display.

The IINCP

Our experiments were run on the MITDG PDP-6/10 using what we have affectionately called our 'interim Interim NCP' (IINCP). Under the IINCP the IMP Interface is treated as a single-user I/O

device which deals in raw network messages. The software
supporting necessary system calls includes little more than the
basic interrupt-handling and buffering schemes to be used later
by the NCP. In short, the user-level programs which brought us
to our historic moments were written close to the hardware with
full knowledge of IMP-HOST Protocol (BBN 1822). When the INCP
and NCP are completed, these programs can be pruned considerably
(80%). The exercise of writing programs which conform to
IMP-HOST Protocol was not at all wasted. Only now can those of
us who are not writing the NCP begin to grasp the full meaning of
RFNM's and their use in flow control. The penalties for ignoring
an impatient IMP, for failing to send NOOPS (NO-OPS) when
starting up, and for blasting data onto the Network without
regard for RFNM's are now well understood.

The Multics Connection

Our quest for historic moments began with the need to
demonstrate that the complex hardware-software system separating
MITDG and Multics was operative and understood. A task force
(Messrs. Bingham, Brodie, Knight, Metcalfe, Meyer, Padlipsky and
Skinner) was commissioned to establish a 'polite conversation'
between a Multics terminal and an MITDG terminal.

It was agreed that messages would be what we call 'network

ASCII messages': 7-bit ASCII characters right-adjusted in 8-bit
fields having the most significant bit set, marking, and padding.
In that Multics is presently predisposed toward line-oriented
half-duplex terminals, it was decided that all transmissions
would end with the Multics EOL character (ASCII <LINE FEED>). To
avoid duplicating much of the INCP in our experiment, the PDP-10
side of the connection was freed by convention from arbitrary
bit-stream concatenation requirements and was permitted to
associate logical message boundaries with network message
boundaries (sic). The 'polite conversation' was thus established
and successful.

Multics, then, connected the conversation to its command
processor and the PDP-10 terminal suddenly became a Multics
terminal. But, not quite:

First, in the resulting MITDG-Multics connection there was
no provision for a remote QUIT, which in Multics is not an ASCII
character. This is a problem for Multics. It would seem that an
ASCII character or the network's own interrupt control message
could be given QUIT significance.

Second, our initial driver program did not provide for
RUBOUT. Because the Multics network input stream bypassed the
typewriter device interface module (TTYDIM), line

3

canonicalization was not performed.   In   a   more   elegant

implementation, line canonicalization could be done at Multics,

providing the type-in editing conventions familiar to Multics

users.   We fixed this problem hastily by having our driver

program do local RUBOUT editing during line assembly, thus

providing type-in editing conventions familiar to MITDG users.

It is clearly possible to do both local type-in editing and

distant-host type-in editing.

        Third, we found that because of the manner in which our

type-in entered the Multics system under the current network

interface (i.e. not through TTYDIM), our remotely controlled

processes were classified 'non-interactive' and thus fell to the

bottom of Multics queues giving us slow response. This problem

can be easily fixed.

The Harvard Connection

        Connecting MITDG terminals to Multics proved to be easy in

that the character-oriented MITDG system easily assembled lines

for the Multics line-oriented system.   We (Messrs. Barker,

Metcalfe) decided, therefore, that it would be worthwhile to

connect the MITDG system to another character-oriented system,

namely Harvard's PDP-10. This move was also motivated by MITDG's

desire to learn more about Harvard's new language systems via

canonicalization   was   not   performed.   In   a   more   elegant

5697

71 122

SOME HISTORIC MOMENTS IN NETWORKING

While awaiting the completion of an interim network control
program (INCP) for the MIT MAC Dynamic Modeling/Computer Graphics
PDP-6/10 System (MITDG), we were able to achieve a number of
'historic moments in networking' worthy of some comment. First,
we were able to connect an MITDG terminal to a Multics process
making it a Multics terminal. Second, we successfully attached
an MITDG terminal to the Harvard PDP-10 System thereby enabling
automatic remote use of the Harvard System for MIT. Third, we
developed primitive mechanisms through which remotely generated
programs and data could be transmitted to our system, executed,
and returned. Using these mechanisms in close cooperation with
Harvard, we received graphics programs and 3D data from Harvard's

MITDG's own consoles.

It was found that Harvard had already provided an ASCII network interface to their system which accepted IMP-Teletype style messages as standard. We quickly rigged up an IMP-Teletype message handler at MITDG and were immediately compatible and connected. But not quite:

First, Harvard runs the Digital Equipment Corporation (DEC) time-sharing system on their PDP-10 which has <control-C> as a QUIT character and <control-Z> as an end-of-file (EOF). MITDG runs the MAC Incompatible Time-sharing System (ITS) which has <control-Z> as a QUIT character and <control-C> as an EOF. This control character mismatch is convenient in the sense that typing <control-C> while connected to Harvard system through MITDG causes the right thing to happen - causes the execution of programs at Harvard to QUIT, as opposed to causing the driver program at MITDG to QUIT. If, however, a Harvard program were to require that an EOF be typed, typing <control-Z> would cause ITS to stop the driver program in its tracks, leaving the Harvard EOF wait unsatisfied and the MITDG-Harvard connection severed.

Second, the Harvard system has temporarily implemented this remote network console interface feature using a DEC style pseudo-teletype (PTY). This device vis-a-vis the DEC system

behaves as a half-duplex terminal which wakes up on a set of
'break characters' (e.g., return, altmode) affording us an
opportunity for an interesting experiment. The use of DDT
(Dynamic Debugging Tool) is thereby restricted (though not
prevented) in that break characters must be scattered throughout
a DDT interaction to bring the PTY to life to cause DDT to do the
right thing. For example, to examine the contents of a core
location one needs to type 'addr/<altmode>' (address slash
altmode) the altmode being only a call-to-action to the PTY. To
alter the contents of the opened location, one must then type
'<rub-out>contents<return>'; the <rub-out> character deletes the
previous action <alt-mode>, the contents are stashed in the open
address, and the <return> signals the close of the address and
PTY wake-up. It would seem that DDT is a program that will
separate the men from the boys in networking.

Third, it was found that the response from the Harvard
system at MITDG was seeemingly as fast as could be expected from
one of their own consoles. This fact is particularly exciting to
those who don't have a feel for network transit times when it is
pointed out that such response was generated through two
time-sharing systems, three user level processes, and three IMPs,
all connected in series.

The Harvard-MIT Graphics Experiment

At Harvard are a PDP-10 Time-sharing System and a graphics oriented PDP-1, both connected to Harvard's IMP. At MITDG are a PDP-6/10 Time-sharing System and an E&S Line Drawing System. It was felt (Messrs. Barker, Cohen, McQuillan, Metcalfe, and Taft) that the time had come to demonstrate that the network could make remote resources available - to give Harvard access to the E&S at MITDG via the network. The protocol for such use of the network was as follows: (1) MITDG starts its network monitor program listening. (2) Harvard starts its PDP-10 transmitting a core image containing an arbitrary PDP-10 program (with an embedded E&S program in this case). (3) MITDG receives the core image from Harvard and places it in its memory at the starting address specified, collecting messages and concatenating them appropriately. (There was no word-length mismatch problem.) (4) Upon collecting a complete image (word count sent first along with starting address), MITDG stashes its own return address in a specified location of the transmitted program's image and transfers control to another image location. (5) Upon getting control at MITDG, the transmitted program executes (in this case sets up and runs an E&S program) and before returning to the MITDG network monitor stashes in specified locations of its image

7

the beginning and ending addresses of its results. (6) With
control returned, the MITDG monitor program then transmits the
results to a listening host which makes good use of them (in this
case a PDP-1 which displays them.). (7) Then the MITDG program
either terminates, returns control back to the image (as in this
case), or waits for more data and/or program. The protocol was
implemented in the hosts and used to run a Harvard-assembled
version of the E&S Aircraft Carrier Program (written originally
by Harvard's Prof. Cohen) at MITDG and to display the resulting
dynamic display on Harvard's PDP-1 driven DEC scopes. The
Carrier Program was 'flown' from MITDG and the changing views
thus generated appeared both at MITDG and Harvard. The picture
was observed to change (being transmission limited) on the order
of twice each second (perhaps less often). But all was not
rosey:

First, it was observed that during the experiment prompting
messages to the IMP-Teletypes were often garbled. Most of the
garbling can be attributed to the ASR-33 itself, some cannot.
There were no errors detected during data trasmissions not
involving the IMP-Teletypes.

Second, during attempts to fly the Carrier from Harvard, we
stumbled across a yet undiagnosed intermittent malfunction of

8

36-Bit Words Transmitted
From Harvard's PDP-10 to
MITDG's PDP-10

-count	origin-1

Image control word.

Image: | start address of results |

Image+1: | end address of results |

Filled in by Harvard's program during its execution.

Image+2: | ------unused-------- |

Image+3: | program stop address |

Image+4: | program start address |

Filled in by MITDG for return of control.

Image+5: | |

Image control word
and image arrive in
network size buffers
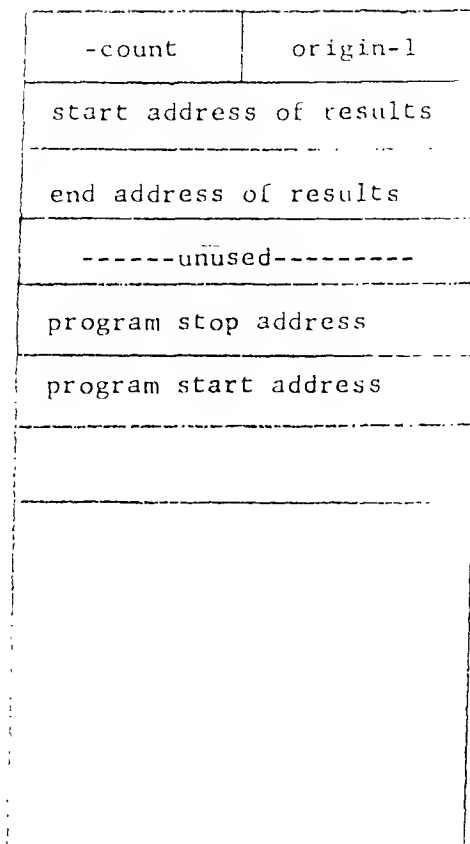which are stripped of
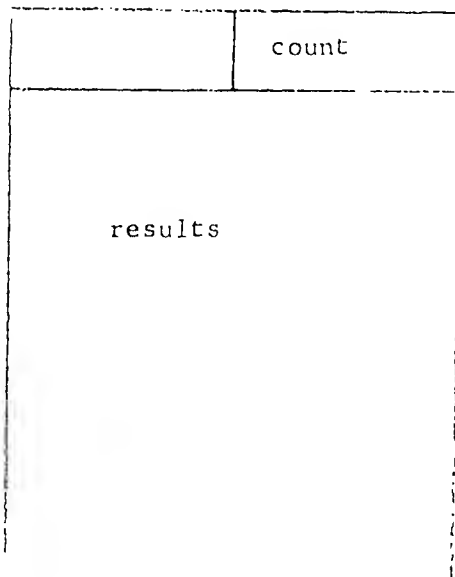marking and padding
and concatenated.

36-Bit Words Transmitted
From MITDG's PDP-10 to
Harvard's PDP-1

	count

First word of results
specified in Image+0.

results

Last word of results
specified in Image+1.

(presumably) the MITDG hardware and/or software which caused our
network connection to be totally shut down by the system during
by-directional transmission. This problem is currently under
investigation.

Third, the response of the total system was slow compared to
that required to do real-time dynamic graphics. One would expect
that if this limitation is to be overcome, higher bandwidth
transmission lines, faster host response to network messages,
and/or perhaps a message priority system will be required.

General Comments

In producing 'network ASCII messages' we were required to
bend over backwards to insert marking so that our last data bit
could fall on a word boundary. Surely there must be a better
way. The double padding scheme and its variants with or without
marking should be considered. Given the current hardware, it
would seem that double padding with marking would be an
improvement. A simple(?) fix to host IMP interfaces enabling
them to send only good data from a partially filled last word
would permit a further improvement: marking and host-supplied
single padding.

In these initial experiments Harvard used the IMP-Teletype
message convention or what we call 'IMP ASCII messages' (without

10

marking) because it would allow them to use IMP-Teletypes for logging in and testing. Multics, on the other hand, used the standard network message format (with marking) to have Host-Host compatibility as per accepted protocols. Both approaches have merit. The IMP-Teletype message format should be changed to conform with the network standard - it should have marking.

Finally, we would like to announce our readiness to participate in experiments which will further extend our confidence and competence in networking, especially experiments which, like the preceding, will have very large returns with relatively small investment.